2x2ine

Issue #3 **October 2016**

zxzine

table of contents

Published by:
Timothy Swenson
swenson_t@sbcglobal.net
swensont@lanset.com

ZXzine is published as a service to the Sinclair ZX81 community. Writers are invited to submit articles for publication. Readers are invited to submit article ideas.

Created using Open Source Tools:

- OpenOffice
- Scribus
- Gimp
- SZ81
- EightyOne

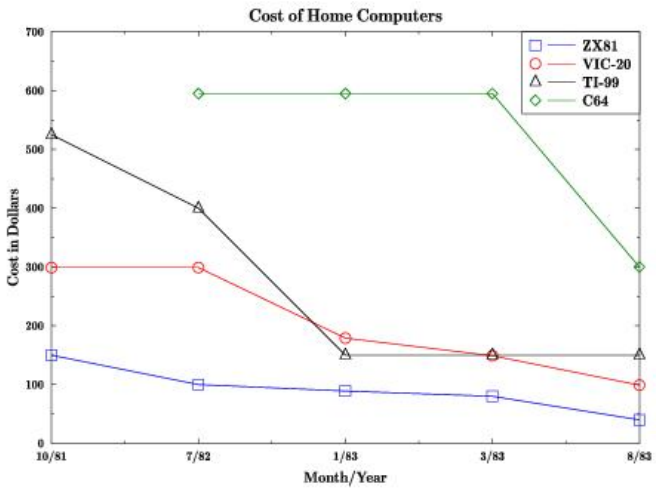
Editorial

Looking at the last issue of ZXzine, I was surprised to notice that it has been a year since the last issue. I've had a number of other projects that have been keeping me from working on the ZX81. I guess I did say that I would get out a new issue when I had enough material. I hope it does not take as long for the next issue.

Home Computer Crash of 1983-85

There a lot of discussion about the Video Game crash of 1983 (which covers 1983 to 1985) in the United States, with a little discussion of how home computers were involved. Not being into video games at the time, I only experienced the crash through the home computers, especially since it caused Timex Computer Company to leave the market. It also left Sinclair with a not-so-good reputation in the United States.

In the crash, a lot of the press went to the fight between Texas Instruments with the TI-99/4A and Commodore with the VIC-20 and C64. Both the VIC-20 and C64 had fairly large profit margins, but the TI-99/4A did not, so with Texas Instruments trying to complete in price with Commodore, left Texas Instruments loosing money.



In 1981, the computers on the market were from Radio Shack, Atari, Commodore, Texas Instruments, and Sinclair. In 1982, the C64 was introduced, with lots of capabilities, but pricey at $595. It has similar capabilities as the 48K ZX Spectrum. Both did rather well as games machines.

In 1983, a bunch more computers hit the market. There was the Coleco Adam, Spectravideo SV-318, and two lower priced systems, the Tandy MC-10 (with single keyword entry like Sinclair) and the Mattel Aquarius, seemly aimed at the pre-teen set. Late in 1983, the T/S 2068 was introduced, a full year and a half after the Spectrum. Hardly any of the system introduced in 1983 really got any share of the market and in less than a year, almost all of them were discontinued. The T/S 2068 was on the market for 4 months before Timex quit. The Aquarius was on the market from June til October.

As computers sat in the market, their priced dropped, and as new models came out the prices dropped even more. The ZX81 hit the market at $150, the lowest price at the time. The VIC-20 was selling for $300 at the same time. When the T/S 1000 came out in July of 1982, the price was dropped to $100. In January of 1983, the T/S 1000 was down to $89, while the VIC-20 was dropped to $179 (since the C64 was already out). Texas Instruments tried to stay in the market by keeping with the price of the VIC-20 and reduced itself from a high of $525 in late 1981 to $150 in Feb. 1983. In the summer of 1983, the T/S 1000 was down to $40 and the VIC-20 was down to $99.

As the price of the more advanced models, like the C64 and the later Atari series, dropped, the price of the earlier system came down even more. With no color or sound, with a very small size, the T/S 1000 was getting the worst treatment. The T/S 1000 was sold mostly at the same retailers that carried Timex watches, like Longs Drugs, Payless Drugs, etc. A few electronic stores carried them, but major retailers like Toys R Us, did not.

Commodore offered a $100 rebate for competitive trade in on any other video game system or home computer. There were even mail-order dealers that would sell a T/S 1000 for $10, along with the C64, so that the buyer could send in the T/S 1000 to use the rebate and save $90.

At this point the T/S 1000 was in fire-sale at all of the retailers, and they did offer the system as low as $10 and $10 for the 16K memory expansion. With that low a price, the general consumer was not thinking of the T/S 1000 as a computer but more as a toy. It was not unusual to ask someone if they owned a computer and here them say, "No, I don't have a computer, ... but I do have this T/S 1000."

Typing in Assembly

One of the key examples of using ROM routines in ZX81 assembly language is getting input from the keyboard and converting it to a character. One of the simplest uses for getting this input is to create a software typewriter, meaning that any character that they user hits on the keyboard is written to the screen, sort of like a typewriter.

The simplest example is type1.asm. In this example, the keyboard is scanned, then converted to a character, which is printed to the screen. Simple, but when run, its main flaw is obvious to see. The scan is much faster than the human finger, so a single depress of a key can result in 8 to 10 characters being printed. If the key is depressed very quickly, then only 3-4 characters are printed.

The next example, type2.asm, a delay routine is put in place to slow down the scan so that only a single character is hit. The problem is that the user can not speed up their typing, but only go at a certain

```
START: LD     A,255
       LD     (count),A     ; Set count to 255
       LD     A,0
       LD     (lstchr),A    ; load lastchar with 0
WAIT:  CALL   KSCAN         ; get a key from the keyboard
       LD     B,H
       LD     C,L
       LD     D,C
       INC    D
       JP     Z, START      ; if no key pressed, then loop, and reset lastchar
       CALL   FINDCHAR      ; Translate keyboard result to character
       LD     A,(HL)        ; Put results into reg A
       LD     HL,lstchr     ; put address of last char into HL
       CP     (HL)          ; is result same as last character?
       JP     Z,REPP        ; How many times have we seen this character?
       CP     $76           ; is result a newline?
       JP     Z,ENDD        ; then exit
       CALL   PRINT         ; Print character
       LD     (lstchr),A    ; load LastChar with current character
       JP     WAIT
REPP:  LD     A,(count)     ; load A with count
       DEC    A             ; dec count
       LD     (count),A     ; store current count in count
       JP     Z,START       ; If count has counted down to 0 then treat as new character
       JP     WAIT          ; Back to wait for a new character
ENDD:
        RET

lstchr:
   DB 00
count:
   DB 00
```

type4.asm

rate. Plus, the delay loop is a waste of compute cycles.

The third example, type3,asm, is a little smarter and based on the idea of a state machine. Simply put, the routine will scan the keyboard, convert it to a character and then see if that character is the same as the last character that was scanned. The first time through the routine, if the key A is hit, then the A character is returned and printed. The A character is stored in the lastchar variable location. When the scan comes round again, if it sees another A, it just ignores it and loops again. Only when no key is scanned will the value of lastchar be reset to 0.

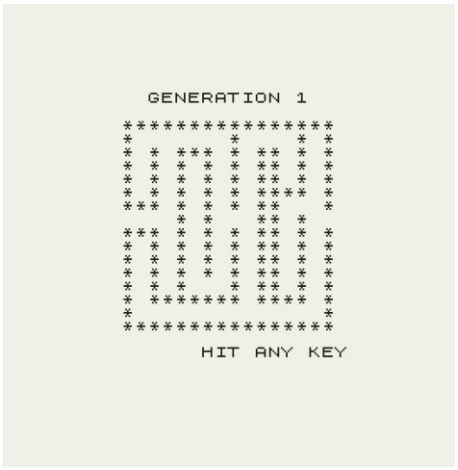
When the user presses the key, this is sensed, the letter is printed and any further scans of the same key are ignored, until the user removed the finger from the key. This removal is noted and lastchar is reset, ready for the next key to be pressed.

The fourth example, type4.asm, takes the third example further. In the third example, there is no key repeat. This means that if the key is held down, only one letter will be printed to the screen, no matter how long the key is pressed. The normal keyboard behavior is to have a key repeat after holding it down for half a second or so. This allows for single key entry when hitting a key and releasing, but when the key is held, the key is then repeated.

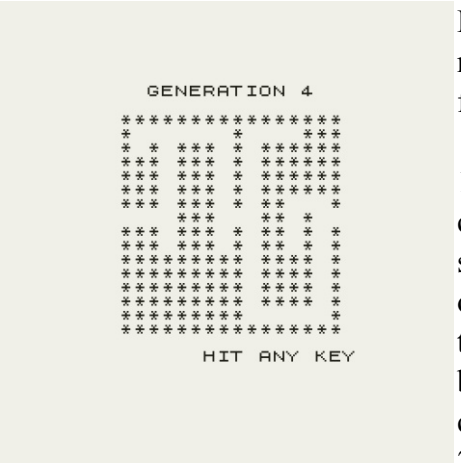
In the fourth example, there is a count of how many times the current key has been seen. Once seen for X times, it is assumed that the user is holding the key down on purpose and then a key reset is done allowing for the key to be printed again. For both the "last character" and the key "count", a memory location is used to hold the values and is basically treated like a variable.

Solving a Maze with Cellular Automata

Past algorithms for solving mazes were brute force approaches to solve the maze just like a mouse. By recursively searching all possible paths the solution will be eventually found. This takes time and lots of memory for stack space. Basem Nayfeh used Cellular Automata to solve the problem with no extra extra memory needed and in a relatively short time.



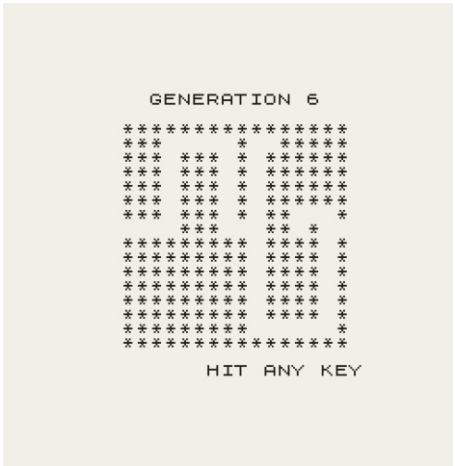
Given a maze that is stored in an array, walls are stored as a cell with a value of 1. The free space between the walls are cells with a value of 0. Each location (cell) in a maze has four directions, like the 4 cardinal directions, East, West, North and South. This means that each cell has a neighborhood of 4 other cells.



Bayeh defines a number of rules for each cell:

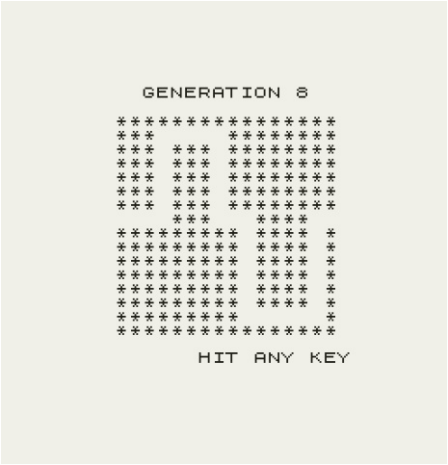
1. If I am a free cell and I'm surrounded by 3 or 4 wall cells, then I will become a wall cell.
2. If I am a wall cell, then I will always remain a wall cell.
3. A free cell surrounded by less than 3 wall cells, remains a free cell.

The last two rules are really not needed as you only need the rule that defines the changes in the value of a cell.



For those that know Cellular Automata, we have the four major components of any cellular automata, the dimension of cells (in this case two dimensions), the state values (1 and 0), the neighborhood, and the rules.

Now the algorithm goes through the array of cells, looking at each cell and the number of neighbors that are wall cells (having a value of 1). When looking at a maze, a number of parts of the maze are dead ends. These dead ends have walls on three sides and one open direction, the one that you came in on. These dead ends will slowly shorten as the dead end is converted to a wall cell. Over time the different dead ends will decrease and keep decreasing until they get to the point where they are gone. The only route left through the maze will be the solution.



The time of the solution is only dependent on the size of the maze and a worse case calculation of what the longest dead end might be. The more complex the maze, having the most dead ends, would be solved faster than one with a few long dead ends.

This implementation of the algorithm shows the maze on each iteration, showing the user how the algorithm works.

ZX81 in Science

When the ZX81 hit the market it was selling for $150 and $100 for the extra 16K. The previous systems on the market were the Apple II selling for $1,350, the Tandy Model 1 was $849 for 16K, the Tandy Model III was $999 for 16K. The Commodore Vic-20 was selling for $299 with 5K.The Atari 400 with 16K was selling for $399. Both the Vic and the Atari needed special cassette recorders selling for between $75 and $90. For those in University, these prices can be rather steep. A 10-year old used small car could be purchased for less than $700.

The appeal of the ZX81 for use in the sciences in University is obvious. It is computer with a built-in language that could be purchased for a reasonable sum. There was no trying to get computer time from the local large systems. The ZX81 could have hardware expansions added cheaply, something that could not be done with the "real" computers in the University.

There were a number of papers submitted to scientific journals using the ZX81 for serious scientific work.

One early paper, published in 1982, is entitled: "Using the Sinclair ZX81 Microcomputer in the Biochemistry Laboratory", by A. G. Booth and A. Dawson of the University of Leeds. The idea of using the ZX81 was based on a book published in 1980 on using pocket programmable calculators in biochemistry. The paper provides two example programs, one on Scatchard Plot and the other on Kinetic analysis.

Another one published in 1982 is "An inexpensive computer and interface for research in the behavioral sciences", by Richard Nicholls and Randall Potter from Clarion State College in Pennsylvania. The paper describes using a ZX81 and a Byte-Back BB-1 control model for controlling a Skinner box, which has a pellet feeder for rewarding an animal subject. They compared the whole cost of a ZX81 system vs other microcontroller-based control systems and found the ZX81 to be a lower cost solution.

A later paper is entitled "A Photoelectric Data Reduction Program for the Sinclair ZX81 Computer" written by Martin West of England, while a visiting student at the Lowell Observatory in the U.S. He mentions the cost of the ZX81 with 16K being only $150, so this was after the first price reduction. The description of the program is to "reduce pulse-count photometric data to the UBW system, optionally calculates extinction and transformation coefficients and zero-point terms." The article details the math behind the calculations used in the program and then includes the code of the program. Obviously the program was meant to be used at the Observatory for professional astronomy.

Another paper is "A potentiometric analyser based on the ZX81 microcomputer" by Batson, Moody and Thomas from the Applied Chemistry

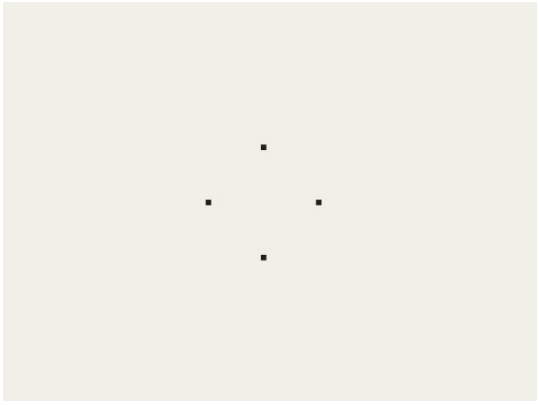
Department of the University of Wales. The ZX81 is used to control a "titration unit" through a parallel bus. A good chunk of the article talks about the design of an analog to digital device. They also describe the software needed to control the hardware attached to the ZX81.

Besides the papers, there is also the book "The ZX81 in science teaching". Published in 1984, toward the heyday of the ZX81, the book has two main topics that apply directly to science, "Computation and mathematical modeling" and "Analogue interfacing". A lot of the book discusses how to do the analog interfacing, including writing machine code and extending the ZX81. It also comes with a number example programs like "Acceleration Tutor", "Fast Timer", "Frequence Metar", Digital Voltmeter", "Radioactive Display", "Wave Motion", etc. Given how late the book came out, I don't know if it was very successful in getting the ZX81 into classrooms. The book authors are Bob Sparks (University of Sterling) and Leon Firth (Paisley College of Technology). My copy of the book actually came from the Paisley College of Technology as a discarded library book.

Circle

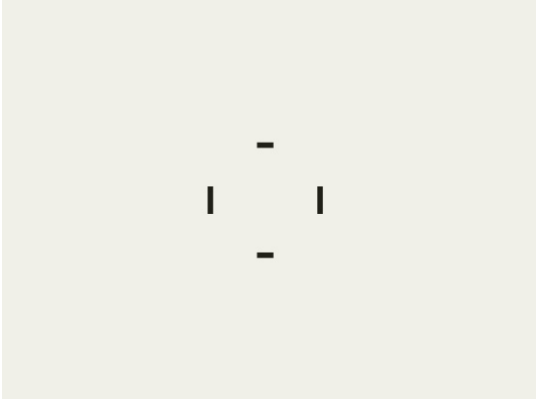
In the last issue the Bresenham algorithm was discussed for drawing a line. The next graphics primitive is drawing a circle. As luck would have it. Bresenham was also involved with optimizing the drawing of a circle. The algorithm is also known as the Midpoint circle algorithm.

What seems the most obvious way to draw a circle is to calculate the location of a point based on the radius from the center of a circle, then using Sin and Cos functions based on the parametric equation for a circle. This method, despite being mathematically sound and what is taught in math class, is a fairly compute intensive algorithm.

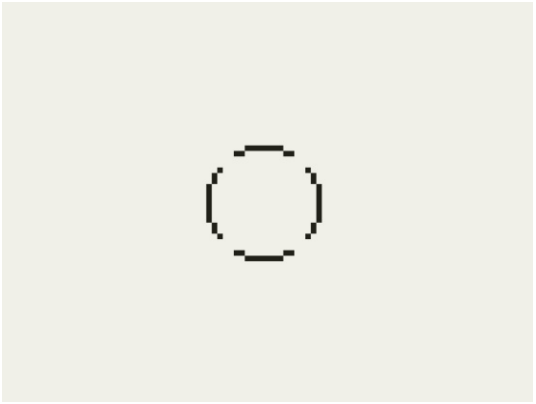


```
# circle.bas
#    Implementation of Bresenham's
       Circle Algorithm
   let x0 = 30
   let y0 = 20
   let rad = 10
   gosub @circle
   stop
@circle:
   let x = rad
   let y = 0
   let err = 1 - x
@cirloop:
   plot x0 + x,y0 + y
   plot x0 - x,y0 + y
   plot x0 + x,y0 - y
   plot x0 - x,y0 - y
   plot x0 + y,y0 + x
   plot x0 - y,y0 + x
   plot x0 + y,y0 - x
   plot x0 - y,y0 - x

   let y = y + 1
   if err < 0 then goto @jump1
   if err >= 0 then goto @jump2
   goto @jump3
@jump1:
   let err = err + 2*y+1
   goto @jump3
@jump2:
   let x = x - 1
   let err = err + 2*(y-x+1)
@jump3:
   if x >= y then goto @cirloop
   return
```



The Besenham algorithm uses only integer math with addition and multiplication, making it a much faster algorithm. The downside is that it does not draw a circle like the second hand sweeping around the circle of a clock. It starts with 4 points of the circle and fills in the gap between the 4 points.

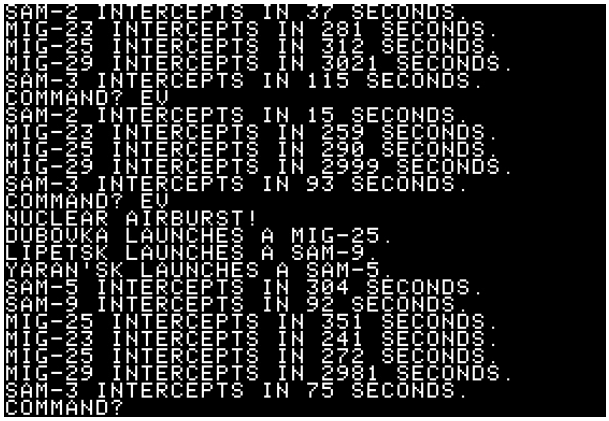


B-1 Bomber for ZX81

Avalon Hill Game Company was a major producer of board games in the United States. They did not do simple family games, but more advance adult-oriented games, and they were most well known for doing wargames, like Squad Leader, Panzer Blitz, Jutland, etc.

In 1980, Avalon Hill started looking into doing computer games, by creating a "Microcomputer Games, Inc." division, looking to convert some of their board games to computer games. One of the first games was "B-1 Nuclear Bomber", first released for the Apple II. Later it came out for the TRS-80, Atari 400/800, CP/M, TI-99/4A, and MS-DOS. This is all well documented in Wikipedia, but what is not mentioned is that it also came out for the ZX81 and the T/S 1000. The January 1985 Avalon Hill Games and Parts Price List has a large table showing their computer games and all of the different systems that they are available for. One column is labeled "Timex/Sinclair" and it can safely assumed that they are referring to the T/S 1000 and not the T/S 2068. The only item listed in the "Timex/Sinclair" column is "B-1 Nuclear Bomber". It is game item 4003 and comes with cassettes for the T/S 1000, the TRS-80 and the TI-99/4A. The cost is listed as $16, which made it one of the cheapest computer game from Avalon Hill.

The early versions of the games was text only and the reviews were actually pretty horrible. The game tied for eighth place in a "Dog of the Year" award from the magazine Softline. I have seen the game on a number of abandonware sites, but I have yet to find the ZX81 version of the game.



DOS version of B-1 Bomber

Atari version of B-1 Bomber